routing Optimization

An exploration of mixed-integer optimization in Python

April 27, 2020

Cameron Cofskey

**Introduction**

This capstone project is an extension of the optimization course taught by Professor Leonardo Lozano. The class explored different ways to solve optimization problems, including both linear and nonlinear methods using manual calculations and mathematical software. One of the topics covered later in the semester was routing optimization, a specific class of problems dedicated to finding the best route between a set of locations, given a certain cost of traveling from one node to another. That cost is typically defined in terms of time or distance, and there are many variations within this type of problem. Determining the best way to move between locations has many applications, including trucking delivery routes that incorporate traffic, inventory or delivery windows. They also have applications in circuit board manufacturing, and perhaps most famously, for a salesman to decide the best order to visit customers on a sales route. Google Maps is one of the more sophisticated examples of this type of problem, and it is used daily by millions of people to determine how they get from a starting location to a destination.

**Motivation**

Because this field has such a wide base of applications, we were only able to scratch the surface of different types of applications, touching mostly on the simplest versions of each problem-solving methodology throughout the semester. However, the final project of the semester presented quite a challenge to the students. It was called the drone problem, and its full details will be described below. Teams were tasked with formulating a model and coding it in a software package called Xpress MP to solve several variations of the problem. Most students had not developed optimization models before this class, and even fewer had coded in Xpress MP. This made the final project an extremely challenging yet rewarding problem to tackle.

After successfully deriving a model, it quickly became apparent that developing a working model for a route including as many as fifty delivery locations would not be feasible for most student’s computers to handle. This in addition to the novelty of the Xpress MP platform left lingering questions of how to attack this problem in a more efficient manner.

Dr. Charles Sox had already begun to use Python in solving simple optimization problems at the undergraduate level, and this combined with the experiences in Dr. Lozano’s optimization class led to the idea of exploring solutions to routing optimization problems in an open-source platform. Dr. Sox provided basic linear programming code examples to help facilitate the translation of ideas from the optimization class into the Python programming language.

**Model Summaries**

In the latter half of the optimization class, the focus shifted from simpler linear problems to more complex scheduling and routing problems, which required variables to be binary (0, 1) or integers. This is where the Traveling Salesman Problem was first introduced.

Terminology

* Objective – the focus of the optimization exercise, usually a maximization or minimization of some set of values (profit, cost, etc.)
* System – the entire realm of factors influencing the problem objective, including variables, parameters and constraints
* Variables – internal factors affecting the objective that can be controlled by one or more decision-makers
* Parameters – external factors affecting the objective that exist independently from decision-makers
* Constraints – factors representing limitations of resources that define the range of possible solution values
* Nodes – locations in a routing optimization problem
* Arcs – possible paths between nodes

Traveling Salesman Problem (TSP)

Problem Overview

The traveling salesman problem is an optimization concept that surfaced as early as the 1830’s, but was not solved mathematically until 1930, and really did not increase in popularity until the 50’s or 60’s. The problem statement is as follows:

**A salesperson must make stops in a certain number of cities. What is the best route for them to take to minimize the distance traveled?**

This problem, if formulated with fewer than, say 10 cities, is nothing more than a computational exercise of rotating through each possible route from one city to the next and choosing the best result. However, as the problem grows beyond even 20 nodes, the problem begins to balloon into a very computationally expensive exercise. This is due to the growth of the problem at a factorial rate, which is at a faster rate than linear, or even exponential growth. Dozens of methodologies have been devised to develop a faster and faster way of coming to this solution. Some of the exact methodologies are still very computationally expensive, while many scientists and mathematicians have resorted to solution approximations and dynamic programming to expand the size of the problems that can be solved.

Solution Methodology

These concepts were used to solve this instance of the traveling salesman problem:

* **Flow in / out** – the number of routes going into a certain node should sum to one, and the number of routes leaving a node should also sum to one.
* **Miller-Tucker-Zemlin (MTZ) formulation** - this method of solving the traveling salesman problem is necessary to eliminate what are called “subtours” on the route. They prevent the salesman from returning to a node already visited by creating an additional decision variable representing the order in which the salesman visits the node. Each time the salesman visits a node, this variable increases by one, and they cannot visit a node with a position value less than the current value. (See Appendix B for more information)

Drone & Truck Problem

Problem Overview

The drone and truck problem is an extension of the traveling salesmen problem with one added layer of complexity. We now think of the salesman as a truck that must make deliveries to various customers. In this scenario we have the option of taking 1 to 2 drones onboard the truck to help with deliveries. The truck and drone have different costs per unit of distance traveled. For the demonstrated examples, the ratio of truck cost to drone cost is 10:3. The goal is to still find the route with minimal costs while choosing to use either the truck or the drone to make a delivery. Since the drone is onboard the truck the drone must launch from a node that the truck is currently located and must return to that node before the truck can move on.

Solution Methodology

* **First Arc** – the sum of all routes leaving the origin must equal one
* **Last Arc** – the sum of all routes arriving at the origin must equal one
* **Flow Balance** – the sum of all routes arriving at an intermediate node minus all routes leaving that same node must equal zero
* **Launch Constraint** – For a drone to leave a node, the truck must have visited that node.
* **Covering Constraint** – Each node must be visited by either a drone or the truck
* **MTZ Truck Constraint** – (see MTZ constraint for traveling salesman)
* **MTZ Truck: Position 1** – the first position variable must equal 1 at the origin

OR-Tools

To solve these problems, we leveraged a package developed by Google to solve optimization problems called OR-Tools. According to Google developers, “OR-Tools is open source software for combinatorial optimization, which seeks to find the best solution to a problem out of a very large set of possible solutions.” The specific solver used for the TSP and drone problems is the [Glop linear optimizer](https://developers.google.com/optimization/lp/glop) for linear and mixed integer programming.

**Modeling**

TSP Model

**Variables & Parameters:**

**Objective function:**

**Constraints:** s.t.

*Flow out:*

*Flow in:*

*MTZ constraint:*

Drone Model

**Variables & Parameters:**

**Objective function:**

**Constraints:** s.t.

*First Arc:*

*Last Arc:*

*Truck Balance:*

*Launch Constraint:*

*Covering Constraint:*

*MTZ truck constraint:*

*MTZ truck position 1:*

**Results**

For simplicity, the traveling salesman and drone problems were developed simultaneously. The origin of the route is at a Euclidean coordinate of (0,0). Each delivery location point is then randomly generated a multitude of times to establish a testbed of problem instances. Cost per distance for the truck or salesman was assumed to be $10 per unit of distance, while cost for the drone was assumed to be $3 per unit of distance. For each number of locations to visit, the solution is displayed below for each D=0 drones (TSP), D=1 drones, and D=2 drones, with the associated route of the truck and drone(s) as they are assigned via the objective function:

5 Nodes

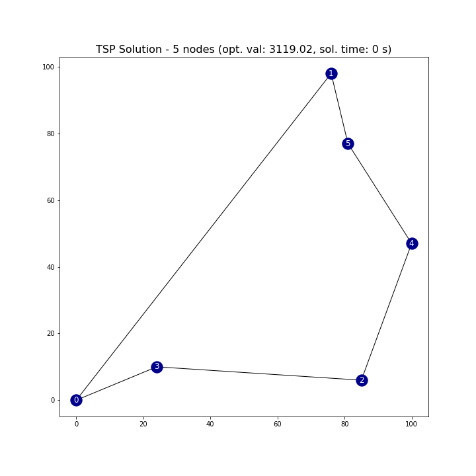
**Figure 1.**

Drones = 0

Orders = 5

Solution Time = < 1s

Cost of Route = $3,119



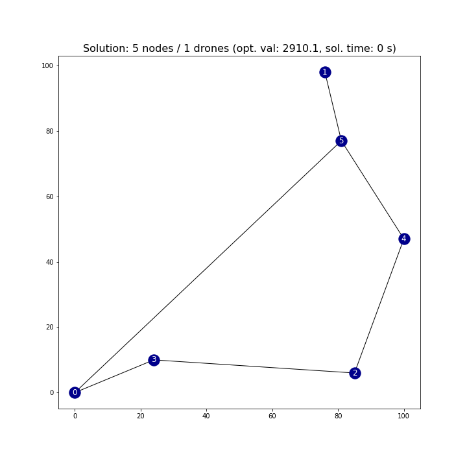
**Figure 2.**

Drones = 1

Orders = 5

Solution Time = < 1s

Cost of Route = $2,910



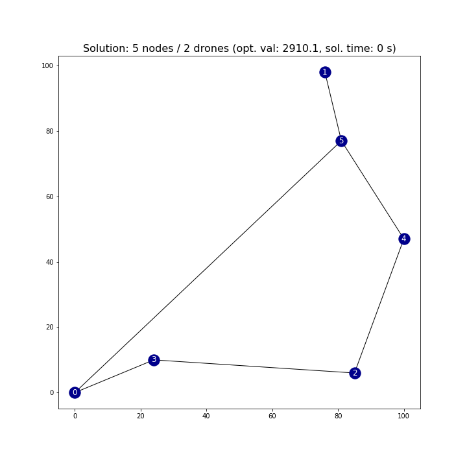
**Figure 3.**

Drones = 2

Orders = 5

Solution Time = < 1s

Cost of Route = $2,910



10 Nodes

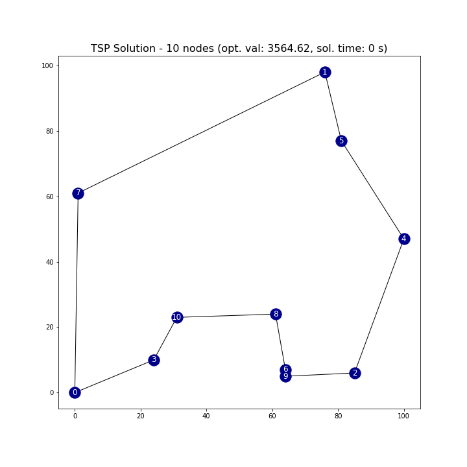
**Figure 1.**

Drones = 0

Orders = 10

Solution Time = < 1s

Cost of Route = $3,565



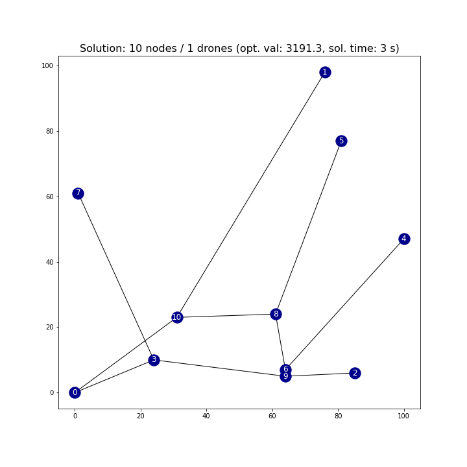
**Figure 2.**

Drones = 1

Orders = 10

Solution Time = 3s

Cost of Route = $3,191



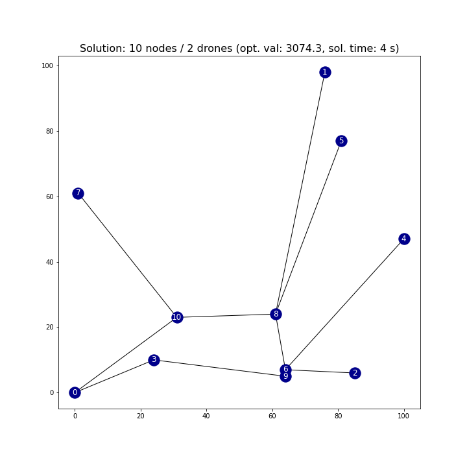
**Figure 3.**

Drones = 2

Orders = 10

Solution Time = 4s

Cost of Route = $3,074



15 Nodes

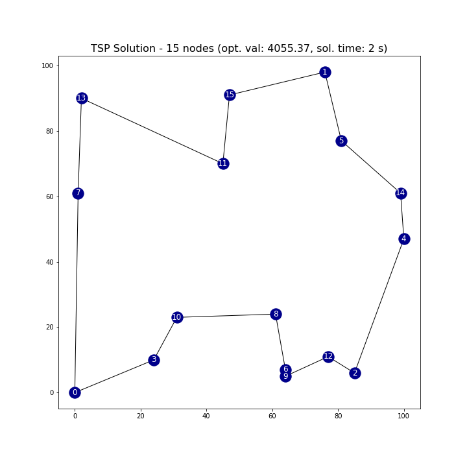
**Figure 1.**

Drones = 0

Orders = 15

Solution Time = 2s

Cost of Route = $4,055



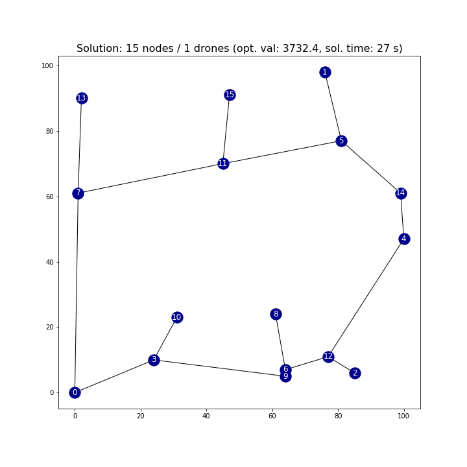
**Figure 2.**

Drones = 1

Orders = 15

Solution Time = 27s

Cost of Route = $3,732



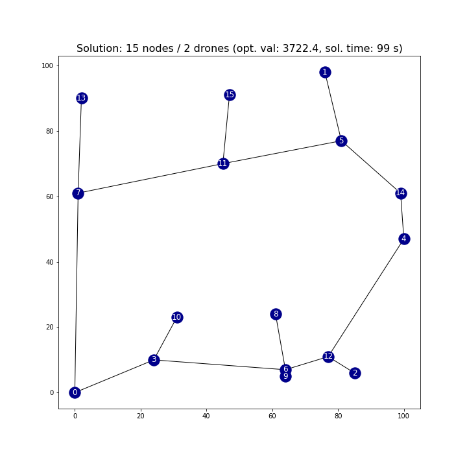
**Figure 3.**

Drones = 2

Orders = 15

Solution Time = 99s

Cost of Route = $3,722



20 Nodes

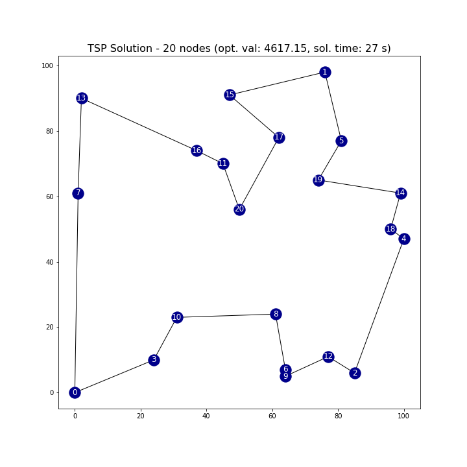
**Figure 1.**

Drones = 0

Orders = 20

Solution Time = 27s

Cost of Route = $4,617



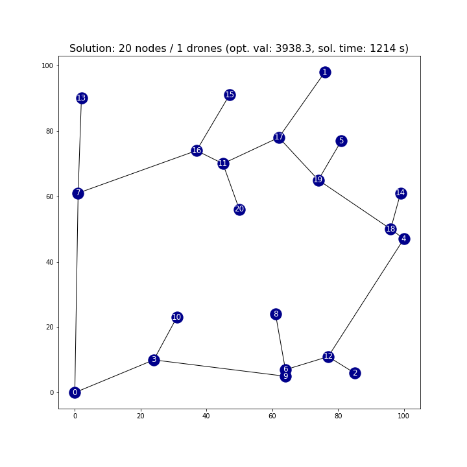
**Figure 2.**

Drones = 1

Orders = 20

Solution Time = 1214s

Cost of Route = $3,938



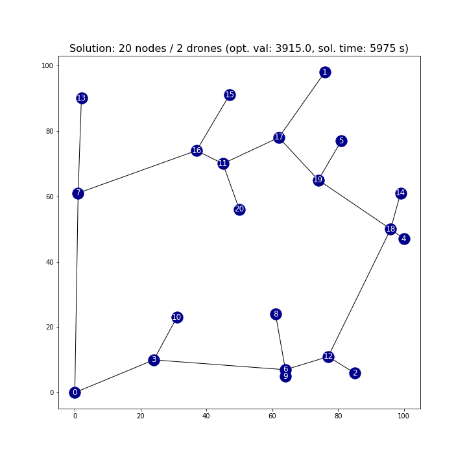
**Figure 3.**

Drones = 2

Orders = 20

Solution Time = 5975s

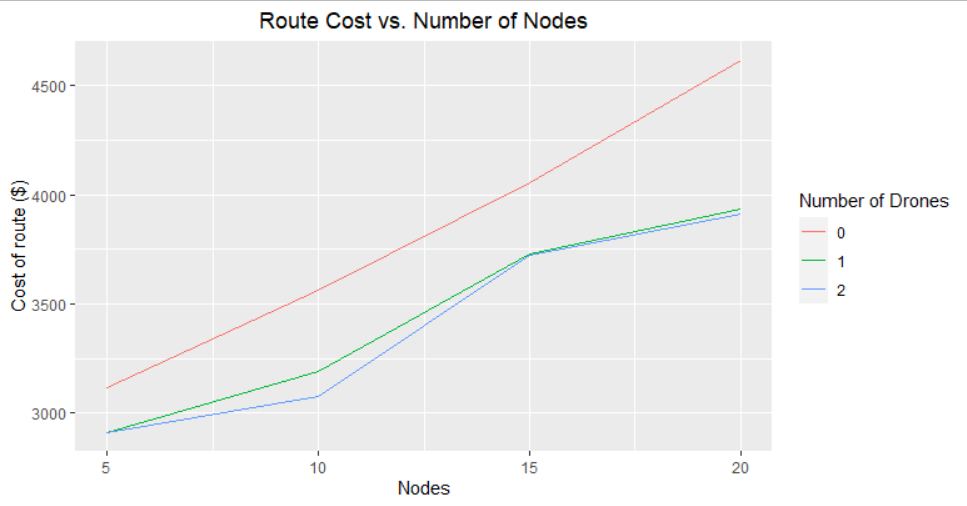
Cost of Route = $3,915



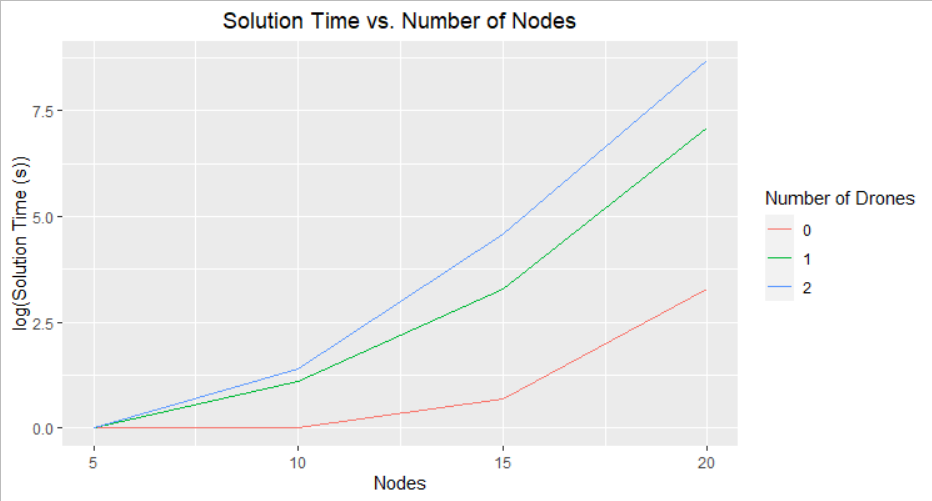
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Model | Nodes | Drones | Variables | Constraints | Cost | Sol. Time (s) |
| TSP | 5 | 0 | 42 | 37 | $ 3,119 | 1 |
| Drone | 5 | 1 | 78 | 48 | $ 2,910 | 1 |
| Drone | 5 | 2 | 78 | 48 | $ 2,910 | 1 |
| TSP | 10 | 0 | 132 | 122 | $ 3,565 | 1 |
| Drone | 10 | 1 | 331 | 191 | $ 3,191 | 3 |
| Drone | 10 | 2 | 331 | 191 | $ 3,074 | 4 |
| TSP | 15 | 0 | 272 | 257 | $ 4,055 | 2 |
| Drone | 15 | 1 | 859 | 479 | $ 3,732 | 27 |
| Drone | 15 | 2 | 859 | 479 | $ 3,722 | 99 |
| TSP | 20 | 0 | 462 | 442 | $ 4,617 | 27 |
| Drone | 20 | 1 | 1762 | 962 | $ 3,938 | 1214 |
| Drone | 20 | 2 | 1762 | 962 | $ 3,915 | 5975 |

Results Summary

Key Takeaways



As this chart indicates, the cost of traveling increases rather linearly as the number of locations increases. However, adding one drone into the equation consistently offers a cost reduction, while the second drone does not add much marginal value above one drone. (Note: as stated above, the cost of the truck is $10 per unit of distance while the drone’s cost is $3)



The solution time is the single most difficult thing about solving these problems combinatorically, as it increases at a factorial rate with the addition of more nodes and constraints. Even with the logarithmic scale used here, it is easy to see the large increase in solution time required for each increase in complexity.

**Further Study**

TSP Heuristics

Since the solution time is the largest obstacle in solving more complex problems, there are many approximate algorithms, or heuristics, that have been employed for faster solutions with results that approach but do not reach optimality. These approaches would be the next stage in exploring solutions for the traveling salesman problem. Some of these approaches are listed below:

* Nearest Neighbor – This approach is of the opinion that the salesman should always travel to the city nearest to them and return once all locations have been visited
* Greedy Heuristic – This creates a path by selecting the shortest path if it does not revisit a node.
* Insertion Heuristics – This approach starts with a subset of the locations to visit and adds the other progressively
* Christofides – This approach guarantees a solution that is no worse than 150% of optimal

Once an initial heuristic has been achieved, there are many other ways to improve these solutions, the best of which is called the Held-Karp lower bound, which is a relaxation of the initial formulation of the TSP.

Dynamic Programming

There are ways in computer science to eliminate redundancies in successive calls to a solver function. This approach has been taken to reduce the traveling salesman problem from an O(n!) solution time to an O(n22n) solution time. This could be something worth exploring since Python is a computer science language first before data science or optimization.

Appendix

**Appendix A: Node Coordinates**

|  |  |  |
| --- | --- | --- |
| Origin | 0 | 0 |
| 1 | 76 | 98 |
| 2 | 85 | 6 |
| 3 | 24 | 10 |
| 4 | 100 | 47 |
| 5 | 81 | 77 |
| 6 | 64 | 7 |
| 7 | 1 | 61 |
| 8 | 61 | 24 |
| 9 | 64 | 5 |
| 10 | 31 | 23 |
| 11 | 45 | 70 |
| 12 | 77 | 11 |
| 13 | 2 | 90 |
| 14 | 99 | 61 |
| 15 | 47 | 91 |
| 16 | 37 | 74 |
| 17 | 62 | 78 |
| 18 | 96 | 50 |
| 19 | 74 | 65 |
| 20 | 50 | 56 |

**Appendix B: A note on the MTZ formulation**

[Introduction to MTZ](http://quanscope.com/TSP_MTZ.pdf)

Sources

“About OR-Tools | Google Developers.” Google, Google, developers.google.com/optimization/introduction/overview.

Joshi, Vaidehi. “Speeding Up The Traveling Salesman Using Dynamic Programming.” Medium, Basecs, 13 Nov. 2017, medium.com/basecs/speeding-up-the-traveling-salesman-using-dynamic-programming-b76d7552e8dd.

Nillson, Christian. Heuristics for the Traveling Salesman Problem. http://160592857366.free.fr/joe/ebooks/ShareData/Heuristics for the Traveling Salesman Problem By Christian Nillson.pdf.

“Travelling Salesman Problem.” Wikipedia, Wikimedia Foundation, 26 Apr. 2020, en.wikipedia.org/wiki/Travelling\_salesman\_problem#History.